Combining DEVS with Multi-agent Concepts to Design and Simulate Multi-models of Complex Systems

Benjamin Camus, Christine Bourjot, Vincent Chevrier

Combining DEVS with Multi-agent Concepts to Design and Simulate Multi-models of Complex Systems

Benjamin Camus, Christine Bourjot, Vincent Chevrier
Université de Lorraine, CNRS, LORIA UMR 7503,
Vandœuvre-lès-Nancy, F-54506, France.
INRIA, Villers-lès-Nancy, F-54600, France
prenom.nom@loria.fr

January 15, 2015

Abstract

We are interested in the multi-modeling and simulation of complex systems, that is representing a complex system as a set of interacting models and simulating it with a co-simulation approach. Representing and simulating the multi-model of a complex system requires to integrate heterogeneity at several levels (representations, formalisms, simulation software, models' interactions...). In this article, we present our approach that consists of combining the DEVS formalism and multi-agent concepts in order to achieve these requirements. The use of the DEVS formalism enables a rigourous integration of models described with heterogeneous formalisms and a rigourous simulation protocol. Multi-agent concepts ease the description of multi-perspective integration and the reuse of existing heterogeneous simulators. We detail the combination of both in the AA4MM approach and illustrate its use in a proof of concept.

Keywords: Complex system, multi-model, meta-model, multi-agent, DEVS

1 Introduction

In this article, we are interested in the design and the study of complex systems. Such systems are characterized by *"a great number of heterogeneous entities, among which local interactions create multiple levels of collective structure and organization"* [4].

Simulation is an important tool for this activity because it allows testing different alternatives and different scenarios while limiting experimentation costs. Most modeling questions about complex systems can only be answered by representing the system as a set of interacting models: a multi-model. Such a multi-model may be heterogeneous as composed of models written in different formalisms, implemented in different

simulation software, and interacting in different ways. The problem is then to integrate this heterogeneity.

Siebert et al. [17] introduced the Agent & Artifact for Multi-Modeling (AA4MM) approach. They proposed multi-agent concepts to describe an heterogeneous multi-model, and they relied on the Discrete EVent System Specification (DEVS) formalism [27] to conceive a decentralized execution algorithm that respects causality constraints. However, this approach considered only one application case: the structural coupling of models for mobile ad-hoc networks study [12]. Within this scope, the co-simulation algorithm was simplified with respect to the DEVS possibilities.

As a consequence, whereas the multi-agent paradigm of AA4MM provides the concepts required to represent a complex system multi-model, the capabilities of DEVS to formalize and simulate such a multi-model were not fully exploited in this approach, and therefore, limit the application domain of AA4MM.

In this article, we present a generalization of this approach to cover the study of other complex systems. We systematically describe the mapping of the DEVS concepts to the AA4MM ones and the implementation of the Chandy-Misra-Bryant algorithm [3] [1] for DEVS within AA4MM.

The following section presents the challenges related to multi-modeling and co-simulation of complex systems. Section "The DEVS formalism for multi-modeling" rapidly positions DEVS as a integration formalism for heterogeneous multi-model. Section "The AA4MM approach" underlines the principles of AA4MM. Section "The AA4MM meta-model" introduces the meta-model of AA4MM and how we map it with DEVS. Section "The AA4MM Simulation Middleware : implementing multi-agent concepts according to the DEVS simulation protocol" details how we translate the parallel conservative DEVS algorithm to specify and implement the AA4MM software. Finally, section "Proof of concept" illustrates with a proof of concept showing the ability of this generalization to describe and simulate an heterogeneous multi-model.

2 Requirements for multi-modeling and multi-simulation

In this section, we detail the key requirements related to complex system multi-modeling and simulation.

Multi-perspective integration: the multi-model may represent the target system with models at different temporal and/or spatial scales, and with different levels of resolution. Such multi-level representation could be needed, for instance, when there is a lack of expressiveness of one level and a second one is required; when available data explicitly refer to different levels of representation; or when the modeling question is explicitly to study the mutual influences between the coupled levels dynamics. Aggregation and disaggregation operations are required to pass from a level to another one [11].

Multi-formalism integration: the representation of a complex system may require the integration of models written in different formalisms [21]. At the execution level, this formalism heterogeneity implies dealing with different scheduling policies: cyclic or variable time-steps, and event-based.

Simulation software interoperability: the models composing the multi-model

may be already implemented in different simulation software. These models and their implementations constitute an expertise that must be capitalized. That's why reusing these models without implementing them again is required. Interoperability processes are required to manage exchange of data between these heterogeneous software. [5]

Dynamic simulation adaptation: dynamically modifying a multi-model may be required during the simulation of a multi-model. For instance, if the models represent different parts of the system's state space [8], or when emerging conditions promote the use of another model than the current used one [25] [24]. Monitoring the simulation, selecting the appropriate model, and switching of model during the simulation is required.

To sum-up, multi-modeling a complex system requires to integrate heterogeneity at several levels (representations, formalisms, simulation software, models' interactions...). In the following section, we investigate the ability of the DEVS formalism to represent and simulate an heterogeneous multi-model.

3 The DEVS formalism for multi-modeling

The DEVS formalism [27] is the most general formalism for discrete event model. An important feature of this formalism is that it can integrate all other formalisms [20].

Within the DEVS formalism, each model of the multi-model is described as a DEVS atomic model. This atomic model which has input and output ports, has a sufficiently generic behavior to represent other formalism ones. The multi-model corresponds to a DEVS coupled model. Figure 1 and table 1 show an example of a multi-model described with a simplified flattened version of DEVS.

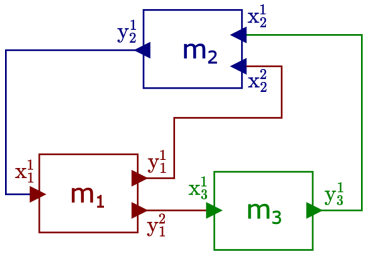

Figure 1: A multi-model composed of three models m_1, m_2 and m_3 as a DEVS coupled model (here described with a simplified flattened version of DEVS).

Once formalized in DEVS, the heterogeneous multi-model can be simulated thanks to the DEVS simulation protocol. This simulation protocol includes parallel and sequential simulation algorithms.

The DEVS formalism explicitly addresses the multi-formalism integration. However, whereas DEVS is compatible with the other requirements of section "Requirements for multi-modeling and multi-simulation", it does not provide explicit solution to:

- models reuse, and the interoperability between their simulators [19] [10];

Descriptions	Notations
Atomic models	$M_d = \{m_1, m_2, m_3\}$
Atomic models's interconnections	$\mathsf{IC} = \{((1,1),(2,2)), ((1,2),(3,1)), ((2,1),(1,1)), ((3,1),(2,1))\}$

Table 1: Formalization of the coupled model of figure 1 in a simplified flattened version of DEVS.

- the use of operations like aggregation or disaggregation, to pass from a perspective of the target system to another one [16];
- the processes of dynamic simulation adaptation. [26]

In following sections, we present the AA4MM approach and how it combines DEVS with multi-agent concepts to fulfill the requirements listed above.

4 The AA4MM approach

AA4MM [17] aims to describe systems as a set of heterogeneous models (namely a multi-model). It proposes a meta-modeling approach based on the multi-agent metaphor to describe a heterogeneous multi-model.

Based on the linguistic levels of [13] and the multi-perspective modelling of [16], we specify the AA4MM's approach as follows (see Figure 2). At the meta-model level, we define a language tailored for the description of heterogeneous multi-models. A multi-model can be then simulated at the simulators level using the AA4MM middleware. This middleware implements the meta-models' concepts and, therefore, can be automatically configured based on a AA4MM multi-model description.

Within the AA4MM meta-model, a multi-model is described with multi-agent concepts. These concepts are combined with the DEVS formalism in order to achieve multi-formalism integration. The concepts are graphically represented (detailed in section "The AA4MM meta-model") and associated with semantic and syntactic constraints guaranteeing a non ambiguous description [18].

At the simulation level, the multi-model is simulated with a co-simulation approach thanks to the AA4MM simulation middleware. Within this middleware, the meta-model's concepts are implemented according to DEVS simulation protocol for coordinating the simulators' execution and managing interactions between models.

5 The AA4MM meta-model

5.1 Generalities

The AA4MM meta-model relies on the multi-agent paradigm to envisage a multi-model as a set of interacting models: each couple model/simulator corresponds to an agent, and the data exchanges between the simulators correspond to the interactions

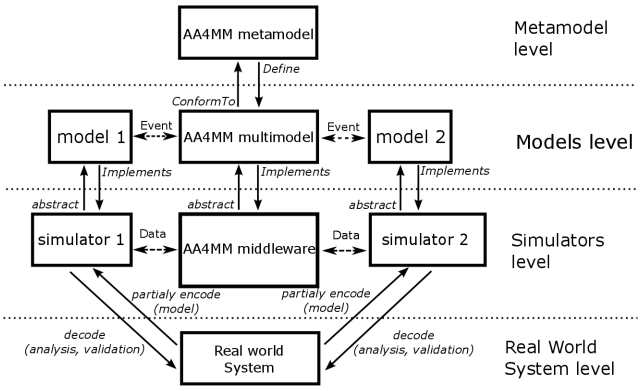

Figure 2: The AA4MM approach.

between the agents. Originality toward other multi-agent multi-model approaches is to consider the interactions in an indirect way within the Agents and Artifacts (A&A) paradigm [15].

Within this paradigm, artifacts support models' interactions as processes outside of the models and express them independently of the models' internal functioning. As a consequence, the simulators interoperability issue is managed by the artefacts. The multi-perspective integration issue is managed as a transformation service of the artefact in charge of the interaction between models.

Moreover, the concept of autonomous agent has been shown [26] to be sufficiently expressive to describe the monitoring of a multi-model and its dynamical adaptation during a simulation.

5.2 Multi-agent Concepts of the meta-model

AA4MM relies on four concepts to describe a multi-model:

- A **model** m_i is a partial representation of the target system implemented in a simulator s_i (symbol in Figure 3d).
- An **m-agent** $\mathcal{A}_i$ manages a model m_i and is in charge of interactions of this model with the other ones (symbol in Figure 3a).
- An interaction from an m-agent $\mathcal{A}_i$ to an m-agent $\mathcal{A}_j$ is reified by a **coupling artifact** $\mathcal{C}_j^i$ (symbol in Figure 3b). A coupling artifact $\mathcal{C}_j^i$ has two roles: for $\mathcal{A}_i$, it is an **output coupling artifact**, whereas for $\mathcal{A}_j$ it is an **input coupling artifact**. The coupling artifacts can transform the data exchanged between the models using operations that can be for instance, spatial and time scaling operations, or aggregation and disaggregation operations [2].
- The **model artifact** $\mathcal{I}_i$ reifies interactions between an m-agent $\mathcal{A}_i$ and its model m_i (symbol in Figure 3c).

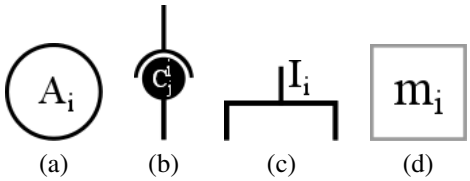

Figure 3: Symbols of the AA4MM components (a) m-agent $\mathcal{A}_i$, (b) coupling artifact $\mathcal{C}_j^i$, (c) model artifact $\mathcal{I}_i$ (d) model m_i.

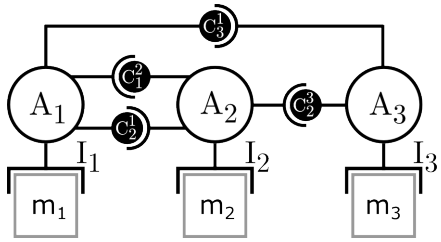

Figure 4: A multi-model described with AA4MM.

Figure 4 is the equivalent of the multi-model of figure 1 described with the AA4MM meta-model.

Different kinds of m-agents with different behaviors exist depending on the models coupling. For instance, co-evolution m-agents coordinate the models execution with other ones, whereas sequential m-agents monitor and dynamically adapt the multi-model during the simulation. In this article, we restrict to the co-evolution m-agent.

In the following section, we present how these concepts are combined with the DEVS formalism in order to manage multi-formalism integration.

5.3 Combining multi-agent concepts with the DEVS formalism

We map the AA4MM multi-model with the DEVS formalism as follows. Each m-agent $\mathcal{A}_i$ sees its model m_i as a DEVS atomic model thanks to its model artefact $\mathcal{I}_i$. Therefore, $\mathcal{I}_i$ acts as a DEVS wrapper [14] for m_i. Each coupling artifact $\mathcal{C}_j^i$ between the m-agents $\mathcal{A}_i$ and $\mathcal{A}_j$ corresponds to an interconnection between the models m_i and m_j.

In AA4MM, an m-agent only has a local knowledge of the coupled model's interconnections. The coupled model's interconnections set IC is split such as an m-agent $\mathcal{A}_i$ only knows:

- which input coupling artifact corresponds to its model's input ports. We define the set of input links IN_i of $\mathcal{A}_i$ as being composed of the couples (j, k) mapping the input coupling artifact $\mathcal{C}_i^j$ with the input port x_i^k.
- which output coupling artifact corresponds to its model's output ports. We define the set of output links OUT_i of $\mathcal{A}_i$ as being composed of the couples (n, j)

mapping the ouput port y_i^n with the output coupling artifact $\mathcal{C}_j^i$.

The connection of the output ports of a model m_i with the input ports of a model m_j is done by the coupling artifact $\mathcal{C}_i^j$.

The link from a model m_i to a model m_j (noted as L_j^i) corresponds to the tuple $(n, k, o_{j,k}^{i,n})$. It maps the output port y_i^n with the input port x_j^k and applies the o_k^n operation to transform the event between these two models representation. By default, an operation corresponds to the identity operation id. Table 2 shows how the coupled model of table 1 is described within AA4MM.

Descriptions	**Notations**
Output links of m_1	$\mathsf{OUT}_1 = \{(1,2),(2,3)\}$
Input links of m_1	$\mathsf{IN}_1 = \{(2,1)\}$
Output links of m_2	$\mathsf{OUT}_2 = \{(1,1)\}$
Input links of m_2	$\mathsf{IN}_2 = \{(1,2),(3,1)\}$
Output links of m_3	$\mathsf{OUT}_3 = \{(1,2)\}$
Input links of m_3	$\mathsf{IN}_3 = \{(1,1)\}$
Links from m_1 to m_2	$\mathsf{L}_2^1 = \{(1,2,o_{2,2}^{1,1})\}$
Links from m_1 to m_3	$\mathsf{L}_3^1 = \{(2,1,o_{3,1}^{1,2})\}$
Links from m_2 to m_1	$\mathsf{L}_1^2 = \{(1,1,o_{1,1}^{2,1})\}$
Links from m_3 to m_2	$\mathsf{L}_2^3 = \{(1,1,o_{2,1}^{3,1})\}$

Table 2: Formalization of the coupled model of figure 1 and Table 1 in the AA4MM formalism.

In the next section, we present how the multi-agent concepts of the AA4MM metamodel are implemented in the AA4MM middleware, according to the DEVS simulation protocol.

6 The AA4MM Simulation Middleware : implementing multi-agent concepts according to the DEVS simulation protocol

In this section we present the operational specification of the AA4MM middleware. This section is articulated as follows. Section "Events communication through the environment" is concerned with m-agents communication through the artifacts. Section "M-agents' coordination through the environment" is concerned with the co-simulation coordination.

6.1 Events communication through the environment

In AA4MM, the environment is the medium of m-agents' communications. According to the A&A paradigm, this environment is composed of artifacts.

A model artifact $\mathcal{I}_i$ contains primitives to manipulate a simulation software. This artifact acts as a DEVS wrapper for the simulator. It implements the function of the DEVS simulation protocol by the following functions.

- $init()$ initalizes the model m_i. It sets the parameters and the initial state of the model.
- $processExternalEvent(e_{in_i}, t_i, x_i^k)$ processes the external input event e_{in_i} at simulation time t_i in the k^{th} input port of m_i, x_i^k.
- $processInternalEvent(t_i)$ processes the internal event of the model m_i scheduled at time t_i.
- $getOutputEvent(y_k^i)$ returns $e_{out_i}^k$, the external output event at the k^{th} output port of m_i, y_k^i.
- $getNextInternalEventTime()$ returns the time of the earliest scheduled internal event of the model m_i.

These functions have to be defined for each simulation software.

The coupling artifact functioning ensures the decentralized events communication between the m-agents.

A coupling artifact $\mathcal{C}_j^i$ works like a mailbox: the artifact has a buffer of events where the m-agents can post their external output events and get their external input events. $\mathcal{C}_j^i$ proposes the $post(e_{out}^k)$ function to $\mathcal{A}_i$. This function stores and transforms (according to $\mathcal{C}_j^i$'s operation) the external output event e_{out}^k of output port y_i^k, in the artifact's buffer. $\mathcal{C}_j^i$ proposes three functions to $\mathcal{A}_j$:

- $getEarliestEvent(k)$ returns the earliest external input event for the k^{th} input port of m_j, x_j^k.
- $getEarliestEventTime(k)$ returns the time of the earliest external event for the k^{th} input port of m_j, x_j^k.
- $removeEarliestEvent(k)$ removes from the artifact's buffer the earliest external event for the k^{th} input port of m_j, x_j^k.

6.2 M-agents' coordination through the environment

According to our multi-agent approach, each m-agent is an autonomous entity. Therefore, the m-agents perform the simulation of a multi-model in a parallel way.

A parallel simulation of a multi-model must respect the causality constraint: each atomic model must process its events (internal and external) in an increasing temporal order [9, 27].

Two types of approaches exist to fulfill the causality constraint [7]:

- **Conservative approaches** consist of insuring that the causality is never broken during the simulation.
- **Optimistic approaches** consist of letting the models execute without taking care of the causality constraint, detecting when the causality constraint is broken and then rolling back the simulation to this point.

The optimistic approaches require all the simulators to have a roll back capability implemented either with a state saving or inverse computation strategy [7]. As this requirement strongly restricts the type of the simulators which can be used, we have made the choice in the current AA4MM specification to take a conservative approach.

The parallel conservative DEVS simulator is based on the principle of the Chandy-Misra-Bryant algorithm [3] [1]. Proofs that this algorithm is deadlock free and respects the causality constraint can be found in [27]. The advantage of this algorithm for the AA4MM approach is that it is fully decentralized. It is then compatible with the multi-agent paradigm of the AA4MM meta-model.

We translate this algorithm in the AA4MM's concepts as follows.

The behavior of each m-agent corresponds to the DEVS conservative parallel simulator's one. Each m-agent $\mathcal{A}_i$ shares in its environment its own Earliest Output Time estimated noted EOT_i. EOT_i corresponds to the date (in simulation time), below which $\mathcal{A}_i$ guarantees it will not send new external output event. Therefore, each coupling artifact $\mathcal{C}_j^i$ can store an EOT (initially equal to 0). $\mathcal{A}_i$ shares its $\mathsf{EOT_i}$ by updating the EOT of its output coupling artifacts.

Each m-agent $\mathcal{A}_i$ uses the EOTs of all of its input coupling artifacts to compute its Earliest Input Time estimated noted EIT_i. This EIT_i corresponds to the date (in simulated time) below which $\mathcal{A}_i$ will not receive any new external input event. EIT_i corresponds to the minimum EOT of all of $\mathcal{A}_i$'s input coupling artifacts. Therefore, each m-agent $\mathcal{A}_i$ accesses the EOT of all of its input coupling artifacts.

For each m-agent $\mathcal{A}_i$, all the events (internal or external) with a timestamp inferior or equal to EIT_i are said to be safe to process. In order to fulfill the causality constraint, each m-agent must process only safe events, and in an increasing timestamp order.

The EOT_i of each m-agent $\mathcal{A}_i$ is equal to the minimum between:

- the date of its model's next internal event nt_i
- the date of EIT_i plus its model's minimum propagation delay D_i.

$$\mathsf{EOT}_i = min(nt_i, EIT_i + D_i)$$

$D_i (D_i > 0)$ corresponds to the minimum delay (in simulated time) below which the processing of an external event can't schedule a new internal event in a model m_i. D_i has to be determined for each model m_i in the multi-model.

To store and access its EOT, each coupling artifact $\mathcal{C}_j^i$ proposes two functions:

- $\mathcal{A}_i$ can use the $setEOT$ to update EOT_i in the artifact.

- $\mathcal{A}_j$ can use the $getEOT$ to get EOT_i.

To execute a conservative simulation of its model m_i, each m-agent $\mathcal{A}_i$ follows this cycle:

1. Get the EOT of each of its input coupling artifact and compute its EIT
2. Process safe events in a temporal increasing order
3. Compute its EOT and update the EOT of its output coupling artifacts.

In order to process safe events in a temporal increasing order, $\mathcal{A}_i$ follows this cycle:

1. Getting the time nt_i of its next internal event ($nt_i \in \mathbb{R}$).
2. Getting the time t_{in_i} of e_{in_i}, the earliest external event of all its input coupling artifacts. ($t_{in_i} \in \mathbb{R}$)
3. Determining the earliest event between the next internal event and e_{in_i}.
4. If this event is safe to process, process it.
5. If this event is an internal event, propagate the resulting output external events to other agents.

We developped a new version of the AA4MM middleware to integrate the new behavior of the m-agents and their coordination with the coupling artifacts.

Implementing a multi-model requires the models' simulators (assumed to exist), their model artifacts and the transformation operations. We detail these aspects in the next section.

7 Proof of concept

In this section, we illustrate the ability of AA4MM to describe and simulate a multi-model composed of different formalisms (with individual-based, equation-based and event-based models), different simulators (models are implemented in Netlogo and ad-hoc simulators) and different perspectives (the models represents the system at different time-scale and at different resolution levels).This proof of concept is inspired from the hybrid traffic modeling of [6].

We want to simulate the car traffic of an highway decomposed into three different parts (Figure 5), each described by a specific model:

- In part 1, the speed is limited to 90 km/h, overtaking is forbidden. This part can be subject to traffic jam. It is described with an individual-based model (see section "The individual-based model of the highway").
- In part 2, the speed is limited to 130 km/h, overtaking is allowed. This part is described by an event-based model m_2 (see section "The event-based model of the highway").

- In part 3, the traffic can be considered as regular. This part is described by an equation-based model m_3 (see section "The equation-based model of the highway").

For demonstration purpose, we consider this highway to be on a toric space: cars that go out of part 3 enter into part 1. The multi-model is described with AA4MM in section "The multi-model of the highway".

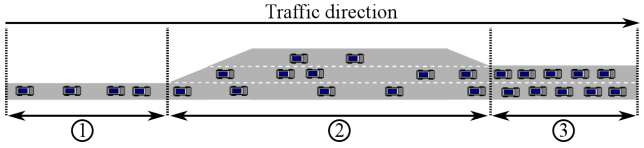

Figure 5: The three parts of the highway.

7.1 The individual-based model of the highway

The individual-based model m_1 is inspired from the traffic model of NetLogo [23] [22]. Within this model, each car is represented individually as an agent[1]. The agents are arranged along an horizontal road. Each agents is described by a position, an orientation (the same for all the agents), and a speed. The behavior of each agent is the following: each time it sees a car too close it decelerates, it accelerates otherwise (up to the speed limit).

The model is based on a cyclic execution: the cars move according to their speeds at each time-step. The model has one output port out_1, and one input port in_1. An external output event sent through out_1 corresponds to the list of the identifiers of cars going out of the road section. An external input event received in in_1 corresponds to a list of the identifiers of cars entering the section.

7.2 The event-based model of the highway

The event-based model m_2 is implemented in an ad-hoc simulator written in Java. This model has three parameters: the length of the road, the minimum and the maximum average speed of a car. Within this model cars are represented individually and described with an identifier and a speed.

It works as follow, when a car enters in the section, its average speed is set randomly according to a given probability distribution between the maximum and minimum average speed of the cars. An internal event corresponds to the exit of a car of the section.

The simulator of the model maintains a stack of the internal event according to the exit date of each cars. The model has one output port out_2, and one input port in_2. An external output event sent through the model's output port out_2 corresponds to a list of the identifiers of cars going out of the section. An external input event received in the model's input port in_2 corresponds to a list of the identifiers of cars entering in the section.

[1] not to be confused with the m-agent of the AA4MM meta-model

7.3 The equation-based model of the highway

The equation-based model m_3 corresponds to the macroscopic model of traffic used in [6]. Within this model, the traffic is described as a flow with a car flow rate, a car density and an average speed. This model is a macroscopic representation of the system compared to the representations of the models m_1 and m_2. This model takes input car flow rate from its input port in_3. The output of the model corresponds to the car flow rate going out of the section.

The model m_3 is implemented in an ad-hoc simulator written in Java. This simulator solves the flow equation by discretizing the simulation time. The simulator is based on a cyclic execution. External input and output events correspond respectively to input and output flow rate. A time step is equal to twenty time steps of m_1.

7.4 The multi-model of the highway

In this section, we detailed how the intuitive multi-model of the highway of figure 6 is described and implemented in AA4MM.

We start by the definition of the model artifacts $\mathcal{I}_1$, $\mathcal{I}_2$ and $\mathcal{I}_3$ for controlling respectively the models m_1, m_2 and m_3. As m_2 is an event-based model, defining $\mathcal{I}_2$'s functions is a straightforward process. For the models m_1 and m_3 which have cyclic time-step scheduling policies:

- the $getNextInternalEventTime$ function of $\mathcal{I}_1$ and $\mathcal{I}_3$ returns the current time of the model plus the duration of a time-step.
- the $processInternalEvent$ function of $\mathcal{I}_1$ and $\mathcal{I}_3$ executes the model for one time-step.
- the $processExternalEvent$ function of $\mathcal{I}_1$ and $\mathcal{I}_3$ sends the external event into the models' input port.
- the $getOutputEvent$ function of $\mathcal{I}_1$ and $\mathcal{I}_3$ collects the external event from the models' output port.

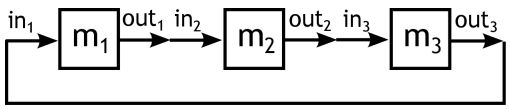

Figure 6: The multi-model of the highway described in an intuitive ambiguous way.

We add three m-agents $\mathcal{A}_1$, $\mathcal{A}_2$, and $\mathcal{A}_3$ for managing the execution of the models m_1, m_2 and m_3.

As m_1 and m_3 have cyclic time-step scheduling policies, external events can not schedule new internal events. Therefore, the delays D_1 and D_3 are equal to $+\infty$. D_2 is equal to the minimum time needed by a car to cross the road, that is to say the length of the road divided by the maximum average speed of a car in m_2.

We add the coupling artefacts $\mathcal{C}_2^1$, $\mathcal{C}_3^2$ and $\mathcal{C}_1^3$ managing the interactions between the models. In order to pass from a micro to a macro representation of the system, we add an aggregation operation agg to $\mathcal{C}_3^2$ transforming a list of cars into a flow rate. To pass from a macro representation of the system to a micro representation, we add a disaggregation operation $disagg$ to $\mathcal{C}_1^3$ transforming flow rate into a list of cars. Figure 7 and table 3 shows how the multi-model of the highway is described using AA4MM.

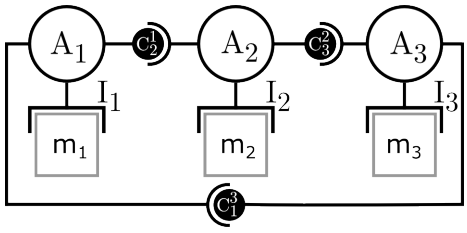

Figure 7: The multi-model of the highway described with AA4MM.

Descriptions	Notations
Output links of m_1	$\mathsf{OUT}_1 = \{(out_1, 2)\}$
Input links of m_1	$\mathsf{IN}_1 = \{(3, in_1)\}$
Output links of m_2	$\mathsf{OUT}_2 = \{(out_2, 3)\}$
Input links of m_2	$\mathsf{IN}_2 = \{(1, in_2)\}$
Output links of m_3	$\mathsf{OUT}_3 = \{(out_3, 1)\}$
Input links of m_3	$\mathsf{IN}_3 = \{(2, int_3)\}$
Links from m_1 to m_2	$\mathsf{L}_2^1 = \{(out_1, in_2, id)\}$
Links from m_2 to m_3	$\mathsf{L}_3^2 = \{(out_2, in_3, agg)\}$
Links from m_3 to m_1	$\mathsf{L}_1^3 = \{(out_3, in_1, disagg)\}$

Table 3: Formalization of the highway multi-model in AA4MM.

To develop this example, we only programmed the code for the interface artifacts and for the operations.

In this proof of concept, we have shown how the multi-agent concepts are used in order to describe an heterogeneous multi-model. Model artifacts manage interoperability of the simulation software with the AA4MM middleware and act as DEVS wrapper. Coupling artefacts' operations manage the integration of macroscopic and microscopic perspectives of the highway. As the middleware is based on DEVS specifications, we can rely on this formalism to manage the parallel simulation of this heterogeneous multi-model.

8 Conclusion

We have presented in this article the mapping of the DEVS formalism and simulation protocol in the multi-agent concepts of the AA4MM meta-model. This combination

of DEVS with multi-agent concepts enables to integrate heterogeneous formalisms, to perform the parallel simulation of the multi-model, to manage simulators interoperability and to integrate multi-perspective in a multi-model.

We have illustrated these possibilities with a proof of concept.

In future works, we plan to take advantage of the expressive power of the A&A paradigm to formalize dynamic adaptation of the multi-model. For instance, continuing with the highway multi-model as a proof of concept, we wish to dynamically detect when the traffic flow is regular or not, and to switch between a macro or a micro model of the road as done in [24].

References

[1] Bryant, R. E. Simulation on a distributed system. In *Proc. of the 16th Design Automation Conference* (1979), 544–552.

[2] Camus, B., Bourjot, C., and Chevrier, V. Multi-level modeling as a society of interacting models. In *Proceedings of the Agent-Directed Simulation Symposium*, ADSS 13, Society for Computer Simulation International (2013), 3:1–3:8.

[3] Chandy, K. M., and Misra, J. Distributed simulation: A case study in design and verification of distributed programs. *Software Engineering, IEEE Transactions on*, 5 (1979), 440–452.

[4] D. Chavalarias, P. Bourgine, E. Perrier, F. Amblard, F. Arlabosse, et al. French Roadmap for complex Systems 2008-2009, 2009.

[5] Dahmann, J. S., Fujimoto, R. M., and Weatherly, R. M. The department of defense high level architecture. In *Proceedings of the 29th conference on Winter simulation*, IEEE Computer Society (1997), 142–149.

[6] El Hmam, M., Abouaissa, Hassane; Jolly, D., and Benasser, A. Macro-micro simulation of traffic flow. In *Proceeding of the 12th IFAC Symposium on Information Control Problems in Manufacturing, INCOM*, vol. 12-1 (2006), 351–356.

[7] Fishwick, P. A. *Handbook of dynamic system modeling*. CRC Press, 2007.

[8] Fishwick, P. A., and Zeigler, B. P. A multimodel methodology for qualitative model engineering. *ACM Trans. Model. Comput. Simul. 2*, 1 (1992), 52–81.

[9] Fujimoto, R. M. Parallel simulation: parallel and distributed simulation systems. In *Proceedings of the 33nd conference on Winter simulation*, WSC '01, IEEE Computer Society (2001), 147–157.

[10] Hu, Y., Xiao, J., Zhao, H., and Rong, G. Devsmo: An ontology of devs model representation for model reuse. In *Proc. of the 2013 Winter Simulation Conference*, WSC '13, IEEE Press (2013), 4002–4003.

[11] Klir J., S. J. Variable resolution modeling in interactive parallel discrete event simulation. In *Electronic computers and informatics.*, K. V. Press, Ed., no. ISBN: 80-8073-150-0 (2004), 353–358.

[12] Leclerc, T., Siebert, J., Chevrier, V., Ciarletta, L., and Festor, O. Multi-modeling and co-simulation-based mobile ubiquitous protocols and services development and assessment. In *7h International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking, and Services*, P. Sénac, M. Ott, and A. Seneviratne, Eds., vol. 73 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Springer Berlin Heidelberg (2010), 273–284.

[13] OMG. *Model Driven Architecture (MDA) Guide*, 2003. OMG doc. ab/2003-06-01.

[14] Quesnel, G., Duboz, R., Versmisse, D., and Ramat, É. DEVS coupling of spatial and ordinary differential equations: VLE framework. In *Proceedings of the Open International Conference on Modeling & Simulation Conference* (2005), 281–294.

[15] Ricci, A., Viroli, M., and Omicini, A. Give agents their artifacts: the A&A approach for engineering working environments in MAS. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS, ACM (2007), 150:1–150:3.

[16] Seck, M. D., and Honig, H. J. Multi-perspective modelling of complex phenomena. *Comput. Math. Organ. Theory 18*, 1 (Mar. 2012), 128–144.

[17] Siebert, J., Ciarletta, L., and Chevrier, V. Agents and artefacts for multiple models co-evolution: building complex system simulation as a set of interacting models. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, International Foundation for Autonomous Agents and Multiagent Systems (2010), 509–516.

[18] Sprinkle, J., Rumpe, B., Vangheluwe, H., and Karsai, G. Metamodelling: state of the art and research challenges. In *Proceedings of the 2007 International Dagstuhl conference on Model-based engineering of embedded real-time systems*, MBEERTS'07, Springer-Verlag (2010), 57–76.

[19] Touraille, L. *Application of Model-Driven Engineering and Metaprogramming to DEVS Modeling & Simulation*. Theses, Université Blaise Pascal - Clermont-Ferrand II, Dec. 2012.

[20] Vangheluwe, H. Devs as a common denominator for multi-formalism hybrid systems modelling. In *Computer-Aided Control System Design. CACSD. IEEE International Symposium on* (2000), 129–134.

[21] Vangheluwe, H., De Lara, J., and Mosterman, P. J. An introduction to multi-paradigm modelling and simulation. In *Proc. AIS2002.* (2002), 9–20.

[22] Wilensky, U. Netlogo traffic basic model. http://ccl.northwestern.edu/netlogo/models/trafficbasic. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL., 1997.

[23] Wilensky, U. Netlogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL., 1999.

[24] Xiong, M., Cai, W., Zhou, S., Low, M. Y.-H., Tian, F., Chen, D., Ong, D. W. S., and Hamilton, B. D. A case study of multi-resolution modeling for crowd simulation. In *SpringSim*, G. A. Wainer, C. A. Shaffer, R. M. McGraw, and M. J. Chinni, Eds., SCS/ACM (2009).

[25] Yilmaz, L., Lim, A., Bowen, S., and Oren, T. Requirements and design principles for multisimulation with multiresolution, multistage multimodels. In *Simulation Conference, 2007 Winter* (2007), 823–832.

[26] Yilmaz, L., and Ören, T. Dynamic model updating in simulation with multimodels: A taxonomy and a generic agent-based architecture. In *In Proceedings of SCSC 2004 - Summer Computer Simulation Conference,* (2004), 3–8.

[27] Zeigler, B., Praehofer, H., and Kim, T. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 2000.